02/15/06

AF / ZIW

OIPE
FEB 1 4 2006
IAP59
PATENT & TRADEMARK OFFICE

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200308340-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s):	**Christopher Henry Rohrs**	**Confirmation No.:**	**5332**
Application No.:	**09/605,271**	**Examiner:**	**Neveen Abel Jalil**
Filing Date:	**06/28/2000**	**Group Art Unit:**	**2165**

Title: Adaptive Type-Partitioned Garbage Collection

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on 1/26/2006.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $500.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

☐ (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d)) for the total number of months checked below:

☐ 1st Month $120　☐ 2nd Month $450　☐ 3rd Month $1020　☐ 4th Month $1590

☐ The extension fee has already been filed in this application.

☒ (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account 08-2025 the sum of $ 500. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

☒ I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450

Date of Deposit: February 14, 2006

OR

☐ I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571)273-8300.

Date of facsimile:

Typed Name:

Signature: Amy Coreau

Respectfully submitted,

Christopher Henry Rohrs

By [illegible]

Robert Plotkin, Esq.

Attorney/Agent for Applicant(s)

Reg No.: 43,861

Date: February 14, 2006

Telephone: (978) 318-9914

Rev 10/05 (AplBrief)

OIPE IAP59
FEB 1 4 2006
PATENT & TRADEMARK OFFICE

ATTORNEY'S DOCKET NO: 200308340-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Christopher Henry Rohrs
Serial No:	09/605,271
Filed:	June 28, 2000
For:	Adaptive Type-Partitioned Garbage Collection
Examiner:	Neveen Abel Jalil
Art Unit:	2165

CERTIFICATE OF MAILING BY "EXPRESS MAIL"

The undersigned hereby certifies that the correspondence listed above is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" Service under 37 CFR § 1.10, postage prepaid, Express Mailing Label No. EV860509659US, in an envelope addressed to Mail Stop Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the 14th day of February, 2006.

Amy Comeau

Amy T. Comeau

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANTS' BRIEF ON APPEAL

This is an appeal pursuant to 35 U.S.C. § 134 from the Examiner's decision rejecting claims 1-26 as set forth in the Final Office Action of November 29, 2005.

02/16/2006 MAHMED1 00000012 082025 09605271
01 FC:1402 500.00 DA

REAL PARTY IN INTEREST

The real party in interest is Hewlett-Packard Development Company, L.P., a Texas Limited Partnership having its principal place of business in Houston, Texas.

RELATED APPEALS AND INTERFERENCES

Applicant's attorney knows of no related pending appeals or interferences.

STATUS OF CLAIMS

Claims 1-26 are pending in this application.

Claims 1-26 stand rejected and are the subject of this appeal. More specifically, claims 1-3, 6-11, 14-19, and 22-26 stand rejected under 35 U.S.C. § 102(e) as being anticipated by Garthwaite (U.S. Pat. App. Pub. No. 2002/0161792 A1). Claims 4, 5, 12, 13, 20, and 21 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Garthwaite in view of Ebrahim et al. (U.S. Pat. No. 5,930,807).

STATUS OF AMENDMENTS

No after-final amendments have been filed in this case.

SUMMARY OF CLAIMED SUBJECT MATTER

All of the claims on appeal include substantially the same relevant limitations related to what is referred to herein as "type-based garbage collection" in an object-oriented computer system. As described in the Background section of the present application, a computer program may request that a portion of a computer's memory be allocated for use by the program (p. 1, lines 9-10). When the allocated portion is no longer needed, a computer program referred to as a "garbage collector" may reclaimed the unneeded memory portion (p. 1, lines 11-12). Conventional "generational" ("age-based") garbage collection collects software objects in memory based on their age (i.e., the amount of time that has passed since they were allocated) (p. 2, line 28 - p. 3, line 8).

In contrast, all of the claims on appeal are directed to performing garbage collection based on the *type* of the software objects being collected. As described in more detail below, an example of an object type is an "integer" type for storing integers, while another example of an object type is a "character" type for storing text. This kind of garbage collection will be referred to herein as "type-based" garbage collection.

More specifically, independent claim 1 is directed to a collector (FIG. 1, element 100; p. 6, lines 5-13) for collecting non-referenced objects stored in a heap (FIG. 1, element 102; p. 6, lines 14-19) by a program (FIG. 1, element 108) executing in a computer system (FIG. 14; p. 24, line 25 - p. 25, line 5).

The collector includes an object allocation routine (FIG. 5A, element 508) which stores (FIG. 6, element 604) an object of a particular type (FIG. 4, p. 8, line 18 - p. 9, line 5) in one of a plurality of logical partitions (FIG. 2, elements 200 and 202; p. 7, lines 3-29) in the heap dependent on a predefined category assigned to the object type (p. 7, lines 16-29; p. 9, lines 19-28), such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition (p. 7, lines 20-29; p. 9, lines 19-28); and

a collection routine (FIG. 5A, elements 500 and 502; p. 9, lines 19-23) which searches one of the logical partitions of the heap for objects to which references are made and reclaims non-referenced objects stored in the searched logical partition of the heap (p. 10, lines 1-14).

Independent claims 9 (apparatus) and 17 (method) include substantially the same limitations as independent claim 1 for purposes of the issues on appeal.

Independent claim 25 is directed to a computer system (FIG. 14; p. 24, line 25 - p. 25, line 5) comprising:

a central processing unit (FIG. 14, element 1408; p. 24, lines 26-27) connected to a memory bus (FIG. 14, element 1406; p. 24, lines 27-28) by a system bus (FIG. 14, element 1412; p. 24, lines 27-28);

an I/O system (FIG. 14, element 1404; p. 25, lines 1-2), connected to the system bus by a bus interface (FIG. 14, element 1414; p. 25, line 29 - p. 26, line 1); and

a collector (FIG. 1, element 100; p. 6, lines 5-13) for collecting non-referenced objects stored in a heap (FIG. 1, element 102; p. 6, lines 14-19) by a program (FIG. 1, element 108) executing in a computer system (FIG. 14; p. 24, line 25 - p. 25, line 5), the collector:

storing (FIG. 6, element 604) an object of a particular type (FIG. 4, p. 8, line 18 - p. 9, line 5) in one of a plurality of logical partitions in the heap (FIG. 2, elements 200 and 202; p. 7, lines 3-29) dependent on a predefined category assigned to the object type (p. 7, lines 16-29; p. 9, lines 19-28), such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition (p. 7, lines 20-29; p. 9, lines 19-28);

searching one of the logical partitions of the heap for referenced objects (FIG. 5A, elements 500 and 502; p. 9, lines 19-23; p. 10, lines 1-14); and

reclaiming non-referenced objects stored in the searched logical partition (FIG. 5A, elements 500 and 502; p. 9, lines 19-23; p. 10, lines 1-14).

Independent claim 26 is a computer program product claim that includes substantially the same limitations as independent claim 25 for purposes of the issues on appeal.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection for review are:

(1) the rejection of claims 1-3, 6-11, 14-19, and 22-26 under 35 U.S.C. § 102(e) as being anticipated by Garthwaite (U.S. Pat. App. Pub. No. 2002/0161792 A1); and

(2) the rejection of claims 4, 5, 12, and 21 under 35 U.S.C. § 103(a) as being unpatentable over Garthwaite (U.S. Pat. App. Pub. No. 2002/0161792 A1) in view of Ebrahim et al. (U.S. Pat. No. 5,930,807).

ARGUMENT

Rejection of claims 1-3, 6-11, 14-19, and 22-26 under 35 U.S.C. § 102(e) (Garthwaite)

Claims 1-3, 6-11, 14-19, and 22-26 stand rejected under 35 U.S.C. § 102(e) as being anticipated by Garthwaite (U.S. Pat. App. Pub. No. 2002/0161792 A1). The Examiner has provided a single ground of rejection, which will be argued in this section by reference to representative claim 1.

Contrary to the assertion of the Final Office Action, Garthwaite does not disclose all of the limitations of claim 1. In general, Garthwaite discloses a *generational* (age-based) garbage collection system. Recall that generational garbage collection decides which software objects to collect (dispose of) based on the *age* of such objects (i.e., the amount of time that has passed since they were created). In contrast, claim 1 of the present application is directed to a *type-based* garbage collection system, which decides which software objects to collect (dispose of) based at least in part on the *type* of such objects. Recall that an example of an object type is an "integer" type for storing integers, while another example of an object type is a "character" type for storing text. Garthwaite does not disclose techniques for performing *type-based* garbage collection. It is therefore clear that claim 1 patentably distinguishes over Garthwaite.

More specifically, Garthwaite does not disclose "an object allocation routine which stores *an object of a particular type* in one of a plurality of logical partitions in the heap dependent on a *predefined category assigned to the object type*" (emphasis added), as required by claim 1 of the present application. As described in the specification of the present application, an example of an "object type" in the Java™ programming language is the char[] object type, which is used to store character strings (p. 8, line 23 – p. 9, line 5). Further examples of object types include "int," "short," "long," and "byte" for storing different types of numerical values (p. 20, lines 25-26).

The Final Office action alleges that Applicant, by pointing to these examples in the specification, has attempted improperly to import limitations from the specification into the claims. This is not correct. Applicant points to the above-referenced text from the specification merely to illustrate examples of object types. It is not necessary, furthermore, to rely on the specification for the meaning of "type" in claim 1 to be clear, because the usage of the word "type" described above is consistent with its well-known meaning in computer science. For example, the *Microsoft Computer Dictionary* (5th ed. 2002) provides the following definition of "type":

> In programming, the nature of a variable – for example, integer, real number, text character, or floating-point number. Data types in programs are declared by the programmer and determine the range of values a variable can take as well as the operations that can be performed on it. *See also* data type.

The same dictionary provides the following definition of "data type":

> In programming, a definition of a set of data that specifies the possible range of values of the set, the operations that can be performed on the values, and the way in which the values are stored in memory. Defining the data type allows a computer to manipulate the data appropriately. Data types are most often supported in high-level languages and often includes types such as real, integer, floating point, character, Boolean, and pointer. How a language handles data typing is one of its major characteristics.

The term "object type," therefore, has a clear meaning in the field of computer science, a meaning that is consistent with its usage both in the specification and the claims of the present application.

Consider again the relevant limitation of claim 1: "an object allocation routine which stores *an object of a particular type* in one of a plurality of logical partitions in the heap dependent on a *predefined category assigned to the object type*" (emphasis added). The specification of the present application discloses examples of assigning a predefined category to an object based on the *type* of that object, and to storing the object in one of a plurality of logical partitions in the heap dependent on the object's predefined category. For example, in one embodiment there are two spaces in the heap in which objects may be stored: hot space and cold space. Some object types may have a preference for being stored in hot space, while other object types may have a preference for being stored in cold space. The preferred space of an object type indicates the category of the object type. For example, the "hot" category is assigned to object types that prefer to be stored in hot space, while the "cold" category is assigned to object types that prefer to be stored in cold space. (*See* p. 7, line 12 - p. 9, line 5.)

Garthwaite does not disclose any assignment of predefined categories to object types, as expressly required by claim 1. Garthwaite does not, therefore, disclose "stor[ing] an object of a particular type in one of a plurality of logical partitions in the heap dependent on a predefined category assigned to the object type," as expressly required by claim 1. Rather, Garthwaite discloses placing "popular" objects in different "cars," where an object is defined as "popular" based on *the number of other objects that reference it* (*see* Garthwaite at p. 8, ¶ 94). Whether an object is "popular," therefore, has no disclosed relationship to the *type* of the object. Even if it is assumed for purposes of argument that "popular" is an example of a "predefined category" within the meaning of claim 1 of the present application, Garthwaite discloses assigning the popular "category" to an object based not on its *type*, but rather on *the number of other objects that reference it*. Garthwaite's categorization of objects as "popular" or "non-popular," therefore, does not anticipate "stor[ing] an object *of a particular type* in one of a plurality of logical partitions in the heap dependent on a predefined category *assigned to the object type*" (emphasis added), as expressly required by claim 1.

The Final Office Action disagrees with this conclusion, stating that "Garthwaite's system organizes objects not only in 'cars' but also according to the class they were initially defined in (Figure 9, 108, type, and Figure 10, 112, class)" (Final Office Action, p. 6, ¶ 6).

This portion of the Final Office Action mischaracterizes the disclosure of Garthwaite. First consider that, in general, Garthwaite discloses a garbage collection system based on the "train algorithm," illustrated generally in FIG. 7 of Garthwaite. In accordance with the train algorithm, "[a] generation to be collected incrementally is divided into sections, which . . . are referred to as 'car sections.' One car section is conventionally collected during one collection cycle." (Garthwaite ¶ [0041].) "Additionally, the car sections are grouped into 'trains,' *which are ordered according to age.*" (Garthwaite ¶ [0042].) Garbage collection generally proceeds according to the ordering of the trains (Garthwaite ¶¶ [0042]-[0052]), making the train algorithm a kind of *generational* (age-based) garbage collection, in contrast to typed-based garbage collection.

Garthwaite's disclosure is directed to "a way of improving" the train algorithm by "employ[ing] collection sets [the set of objects examined during a single garbage collection cycle] that can include more than one car section" (Garthwaite ¶ [0042]). Garthwaite's disclosure, in other words, builds on the age-based train algorithm and does not fundamentally alter its age-based nature.

More specifically, FIG. 9 of Garthwaite illustrates data structures "represent[ing] the type of information a [garbage] collector may maintain in support of the train algorithm"

(Garthwaite ¶ [0090]). These structures include data describing trains and the cars they contain. For example, they indicate the order (sequence) of cars within a train. (Garthwaite ¶¶ [0090]-[0091].)

In particular, structure 100 in FIG. 9 is associated with a portion of memory in the heap. The memory portion managed by structure 100 may contain one car or multiple cars (Garthwaite ¶ [0093]). Although the structure 100 shown in FIG. 9 of Garthwaite includes a field 108 labeled "type," Garthwaite does not disclose storing objects in different logical partitions of the heap based on predefined categories assigned to the "type" stored in field 108. Rather, the "type" field 108 shown in FIG. 9 of Garthwaite merely indicates whether the associated memory section in the heap stores a single car section or multiple car sections (Garthwaite ¶ [0093]). The Examiner has not pointed to any disclosure by Garthwaite that a memory sections associated with a particular value of the "type" field 108 is stored in one of a plurality of logical partitions in the heap dependent on a predefined category assigned to that value of the "type" field 108. Therefore, even if it is assumed for purposes of argument that the "type" field 108 of Garthwaite stores an "object type" within the meaning of claim 1, Garthwaite does not disclose an express limitation of claim 1.

Furthermore, Garthwaite does not disclose another express limitation of claim 1, namely "a collection routine which searches one of the logical partitions of the heap for objects to which references are made and reclaims non-referenced objects stored in the search logical partition of the heap." Recall that the Final Office Action asserts that the embodiment illustrated in FIG. 10 of Garthwaite "organizes objects not only in 'cars' but also according to the class they were initially defined in" (Final Office Action ¶ 6). The following discussion will assume, for purposes of argument, that object "class" is an example of "object type" within the meaning of claim 1.

Even if it is further assumed for purposes of argument that the embodiment illustrated in FIG. 10 discloses the object allocation routine of claim 1, neither this embodiment nor any other portion of Garthwaite discloses "a collection routine which searches one of the logical partitions of the heap for objects to which references are made and reclaims non-referenced objects stored in the searched logical partition of the heap," as required by claim 1.

The Final Office Action appears to assert that the "car-section region" 119 shown in FIG. 10 of Garthwaite is a "logical partition in the heap" within the meaning of claim 1 of the present application. Even assuming for purposes of argument that this is correct, Garthwaite does not disclose a collection routine which searches the car-section region 119 for objects to which references are made and reclaims non-referenced objects stored in the searched logical partition of the heap, as would be required for Garthwaite to anticipate claim 1 under the posited interpretation of Garthwaite.

Rather than disclosing a garbage collection algorithm that searches the *car-section region 119* of FIG. 10 for objects to which references are made, Garthwaite uses the conventional technique of searching through *trains* for objects to which references are made. The Final Office Action effectively admits this, by pointing to ¶ [0063] of Garthwaite as support for Garthwaite's anticipation of the "collection routine" limitation, since ¶ [0063] is simply the conclusion of a discussion of how conventional train algorithms search sequentially through cars in each train to perform garbage collection. Garthwaite does not purport to disclose new techniques for *searching* for unreferenced objects, but rather only to disclose techniques for *storing* objects. For example, Garthwaite claims that the car-section region 119 of FIG. 10 enables cars to be *stored* more efficiently, but does not disclose using the car-section region 119 as a logical partition in the heap that is *searched* for objects to which references are made.

In summary, Garthwaite does not disclose at least one express limitation of claim 1 of the present application. Claim 1, therefore, patentably distinguishes over Garthwaite. All of the other claims subject to the rejection discussed in this section include, either directly or indirectly, the same or substantially the same relevant limitations as claim 1, and therefore patentably distinguish over Garthwaite for at least the same reasons.

Rejection Claims 4, 5, 12, and 21 under 35 U.S.C. § 103(a) (Garthwaite in view of Ebrahim)

Claims 4, 5, 12, 13, 20, and 21 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Garthwaite (U.S. Pat. App. Pub. No. 2002/0161792 A1) in view of Ebrahim et al. (U.S. Pat. No. 5,930,807). All of these claims are dependent claims, which therefore incorporate at least the same limitations as those which form the basis for the arguments in the previous section. The Final Office Action draws on Ebrahim for its alleged disclosure of a write barrier elimination routine, but does not assert that Ebrahim disclosures or suggests any subject matter that upsets the conclusions drawn in the previous section. In fact, neither Garthwaite nor Ebrahim, either singly or in combination, teaches or suggests all of the limitations of the rejected claims, for the reasons provided in the previous section. Claims 4, 5, 12, 13, 20, and 21 therefore patentably distinguish over the combination of Garthwaite and Ebrahim for at least the reasons described above.

CONCLUSIONS

The Examiner's rejections of claims 1-26 should be reversed for the reasons stated above.

If this Brief is not considered timely filed and if a request for extension of time is otherwise absent, applicant hereby requests any extension of time. Please charge any fees or make any credits, to Deposit Account No. 08-2025.

Respectfully submitted,

[illegible] 2/14/2006

Robert Plotkin, Esq. Date
Reg. No. 43,861

Robert Plotkin, P.C.
91 Main Street, Suite 204
Concord, MA 01742-2527
Tel: (978) 318-9914
Fax: (978) 318-9060

APPENDIX A: CLAIMS ON APPEAL

Claim 1. A collector for collecting non-referenced objects stored in a heap by a program executing in a computer system comprising:

an object allocation routine which stores an object of a particular type in one of a plurality of logical partitions in the heap dependent on a predefined category assigned to the object type, such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition; and

a collection routine which searches one of the logical partitions of the heap for objects to which references are made and reclaims non-referenced objects stored in the searched logical partition of the heap.

Claim 2. The collector as claimed in Claim 1 further comprising:

a sample and partition routine which defines a category of an object stored in the heap to be hot or cold.

Claim 3. The collector as claimed in Claim 2 wherein upon determining that a hot logical partition is full, the collection routine searches a cold logical partition and the hot logical partition for referenced objects and moves referenced objects of the hot category stored in the hot logical partition to the cold logical partition.

Claim 4. The collector as claimed in Claim 2 wherein the sample and partition further comprises:

a write barrier elimination routine, which eliminates a write barrier for an intergenerational pointer between an object stored in a hot logical partition and an object stored in a cold logical partition.

Claim 5. The collector as claimed in Claim 4 wherein the write barrier elimination routine eliminates a write barrier by replacing a write barrier machine code instruction with a no operation machine code instruction.

Claim 6. The collector as claimed in Claim 2 wherein the sample and partition routine defines the object category dependent on object type mortality.

Claim 7. The collector as claimed in Claim 6 wherein the sample and partition routine estimates the object mortality dependent on difference of the number of bytes of the object type stored in the heap before a collection and the number of bytes of the object type stored in the heap after the collection.

Claim 8. The collector as claimed in Claim 2 wherein the sample and partition routine partitions the heap to minimize intergenerational pointers between a hot logical partition and a cold logical partition.

Claim 9. A collector for collecting non-referenced objects stored in a heap by a program executing in a computer system comprising:

means for storing an object of a particular type in one of a plurality of logical partitions in the heap dependent on a predefined category assigned to the object type, such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition;

means for searching one of the logical partitions in the heap for referenced objects; and

means for reclaiming non-referenced objects stored in the searched logical partition.

Claim 10. The collector as claimed in Claim 9 further comprising:

means for partitioning the heap into a cold logical partition and a hot logical partition by defining a category of an object stored in the heap to be hot or cold.

Claim 11. The collector as claimed in Claim 10 wherein upon determining that a hot logical partition is full, the means for searching searches a cold logical partition and the hot logical partition for referenced objects and moves referenced objects stored in the hot logical partition to the cold logical partition.

Claim 12. The collector as claimed in Claim 10 wherein the means for partitioning further comprises:

means for eliminating a write barrier for an intergenerational pointer between an object stored in the hot logical partition and an object stored in the cold logical partition.

Claim 13. The collector as claimed in Claim 12 wherein the means for eliminating a write barrier replaces write barrier machine code instructions with no operation machine code instructions.

Claim 14. The collector as claimed in Claim 10 wherein the means for partitioning defines a hot object dependent on object type mortality.

Claim 15. The collector as claimed in Claim 14 wherein the means for partitioning estimates the object mortality dependent on difference of the number of bytes of the object type stored in the heap before a collection and the number of bytes of the object type stored in the heap after the collection.

Claim 16. The collector as claimed in Claim 10 wherein the means for partitioning partitions the heap to minimize intergenerational pointers between the hot logical partition and the cold logical partition.

Claim 17. A method for collecting non-referenced objects stored in a heap by a program executing in a computer system comprising the steps of:

storing an object of a particular type in one of a plurality of logical partitions in the heap dependent on a predefined category assigned to the object type, such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition;

searching one of the logical partitions of the heap for referenced objects; and

reclaiming non-referenced objects stored in the searched logical partition.

Claim 18. The method as claimed in Claim 17 further comprising the step of:

partitioning the heap into a cold logical partition and a hot logical partition by defining hot logical partition objects and cold logical partition objects.

Claim 19. The method as claimed in Claim 18 wherein upon determining that the hot logical partition is full, the step of reclaiming further comprises the step of:

moving referenced objects stored in the hot logical partition to the cold logical partition.

Claim 20. The method as claimed in Claim 18 wherein the step of partitioning further comprises the step of:

eliminating a write barrier for an intergenerational pointer between an object stored in the hot logical partition and an object stored in the cold logical partition.

Claim 21. The method as claimed in Claim 20 wherein the step of eliminating a write barrier replaces write barrier machine code instructions with no operation machine code instructions.

Claim 22. The method as claimed in Claim 18 wherein the step of partitioning further comprises the step of:

identifying a hot object dependent on object type mortality.

Claim 23. The method as claimed in Claim 22 wherein the step of identifying estimates the object type mortality dependent on difference of the number of bytes of the object type stored in the heap before a collection and the number of bytes of the object type stored in the heap after the collection.

Claim 24. The method as claimed in Claim 18 wherein the step of partitioning partitions the heap to minimize intergenerational pointers between the hot logical partition and the cold logical partition.

Claim 25. A computer system comprising:

a central processing unit connected to a memory bus by a system bus;

an I/O system, connected to the system bus by a bus interface; and

a collector for collecting non-referenced objects stored in a heap by a program executing in a computer system, the collector:

storing an object of a particular type in one of a plurality of logical partitions in the heap dependent on a predefined category assigned to the type, such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition;

searching one of the logical partitions for referenced objects; and

reclaiming non-referenced objects stored in the searched logical partition.

Claim 26. A computer program product for collecting non-referenced objects stored in a heap by a program executing in a computer system, the computer program product comprising a computer usable medium having computer readable program code thereon, including a program code which:

stores an object of a particular type in one of a plurality of logical partitions in the heap dependent on a predefined category assigned to the type, such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition;

searches one of the logical partitions for referenced objects; and

reclaims non-referenced objects stored in the searched logical partition.

APPENDIX B: EVIDENCE

No evidence is submitted in support of this Appeal Brief.

APPENDIX C: RELATED PROCEEDINGS

Applicant's attorney knows of no related pending appeals or interferences.